



Creación de Aplicaciones Python

Lección 3: Conocimientos mínimos generales [3/3]

ÍNDICE

Lección 3. – Conocimientos mínimos generales [3/3]	2
Presentación y objetivos.....	2
1. HTML5: breve introducción.....	3
2. CSS3: brevísima introducción.....	12
3. Plantillas Django	16
4. Puntos claves.....	23

Lección 3. – Conocimientos mínimos generales [3/3]

PRESENTACIÓN Y OBJETIVOS

Los Frameworks para programar aplicaciones web con Python nos permiten ahorrar mucho tiempo, y, a su vez, nos permiten realizar cosas sin necesidad de conocer tantas tecnologías.

No obstante, necesitaremos conocer HTML y/o CSS para algunos de estos Frameworks.

Trataremos de explicar HTML con la idea de poder entender las plantillas que se van a usar en Django, por ejemplo.

En el caso de CSS se usa en algunos Frameworks para mejorar la visualización, apariencia de la aplicación.

Para explicar HTML5 lo que haremos será usar el propio ATOM,

No obstante, podría abrirse un bloc de notas y guardar con extensión .html el documento, el resultado sería el mismo.



Objetivos

- Hacer una introducción a HTML
- Hacer una introducción a CSS
- Breve explicación plantillas Django.

1. HTML5: BREVE INTRODUCCIÓN

Nuevo HTML

Tal y como indicamos estaremos usando ATOM para desarrollar código en la mayor parte del curso.

Lo primero es irnos al Entorno y crear un nuevo archivo. (File->New File)

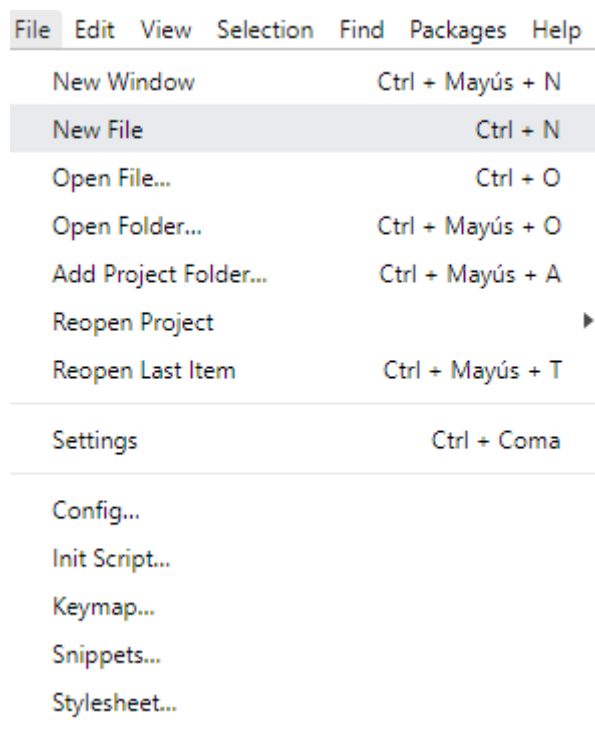


Figura 1.1: Creación del nuevo html

Y, para poder utilizar el IDE lo que haremos será guardarlo a continuación. (Save As..)

Se selecciona una ruta en el PC (a elección de cada uno/a) y se le indica lo siguiente:

Nombre (a elección de cada uno/a nuevamente)

La extensión debe ser ".html", eso sí que es importante.

Tipo: All Files(*.*) esto se selecciona así.

A screenshot of a file dialog box. It has two fields: 'Nombre:' with the text 'web1.html' and a dropdown arrow, and 'Tipo:' with the text 'All Files (*.*)' and a dropdown arrow.

Figura 1.2: Extensión .html

Lo que haremos a continuación será escribir código HTML inicialmente, para ver algunos conceptos básicos.

Lo 1º que podemos indicar es que las "etiquetas de apertura" se escriben entre los símbolos (<....>) y las "etiquetas de cierre" tienen añadida una barra, un "Slash" (</....>)

A screenshot of a code editor window titled 'web1.html'. The editor has a dark background and shows two lines of code: line 1 contains '<html>' and line 4 contains '</html>'. Line numbers 1 through 5 are visible on the left side of the editor.

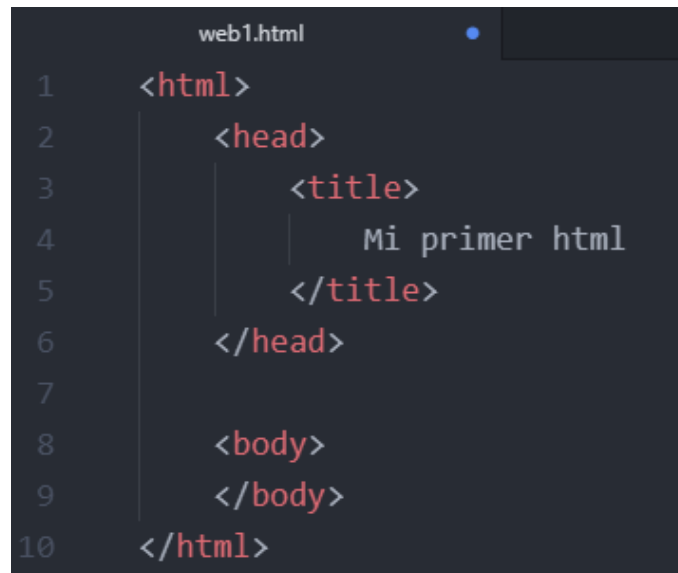
Figura 1.3: Abrir y cerrar etiqueta html

Si nos vamos al lugar donde guardamos el archivo, veremos lo siguiente:



En el cual, si hacemos doble click veremos lo que programemos con nuestro editor de texto (por el momento nada).

En el código HTML suele diferenciarse: un "head" y un "body", podemos añadir un título y hacer muchas cosas más.



```
web1.html
1  <html>
2    <head>
3      <title>
4        Mi primer html
5      </title>
6    </head>
7
8    <body>
9    </body>
10 </html>
```

Figura 1.4: El título del html

Cuyo resultado será el siguiente:

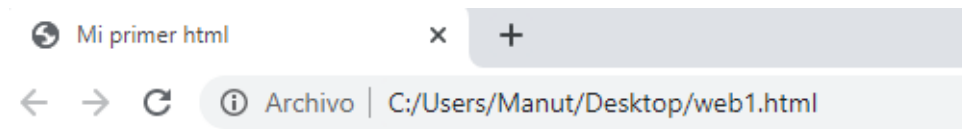


Figura 1.5: Web con título

Por el momento no hemos hecho nada, solo escribir "Mi primer título" (parte superior izquierda).

bgcolor en HTML y <p>

A continuación haremos alguna pequeña demostración más.



```

File Edit View Selection Find Packages Help
web1.html
1  <html>
2      <head>
3          <title>
4              Mi primer html
5          </title>
6      </head>
7
8      <body bgcolor="blue" text="white">
9          <p>Texto 1: Estamos en el body del HTML</p>
10         <p>Texto 2</p>
11         <p>Texto 3</p>
12         <p>Texto 4</p>
13     </body>
14 </html>
    
```

Figura 1.6: Algunos cambios en el body

Si guardamos cada vez que hacemos un cambio y "refrescamos" /actualizamos /Pulsamos "Save" o lo que es más rápido y práctico: CTRL+S, veremos lo siguiente.

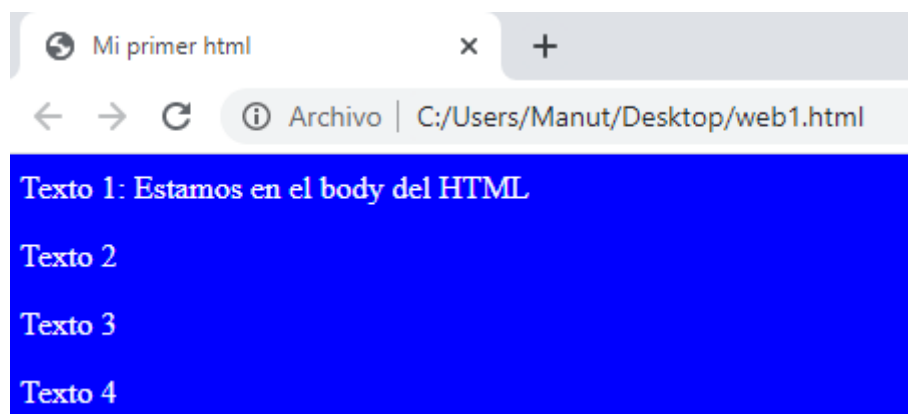


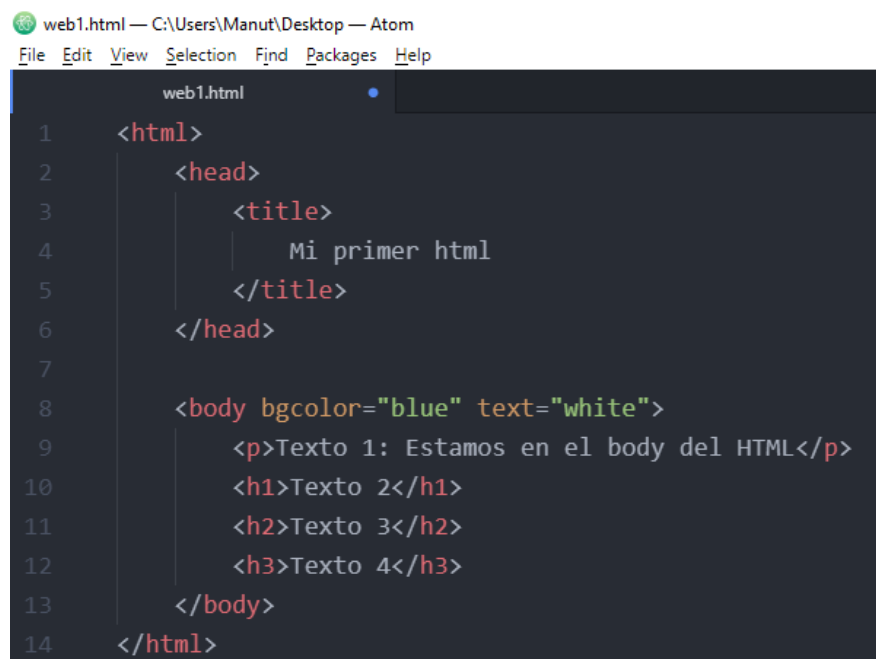
Figura 1.7: Resultado a los cambios en el body

No entraremos en el diseño, ni por el momento tampoco en otras opciones que podemos usar, CSS.

Simplemente en ver que podemos ir escribiendo cosas y cambiando el diseño.

Veremos cómo podemos modificar el tamaño de letra.

Nota: H1 sería el tamaño más grande y 6 el más pequeño.



```
web1.html — C:\Users\Manut\Desktop — Atom
File Edit View Selection Find Packages Help

web1.html
1 <html>
2   <head>
3     <title>
4       Mi primer html
5     </title>
6   </head>
7
8   <body bgcolor="blue" text="white">
9     <p>Texto 1: Estamos en el body del HTML</p>
10    <h1>Texto 2</h1>
11    <h2>Texto 3</h2>
12    <h3>Texto 4</h3>
13  </body>
14 </html>
```

Figura 1.8: h1 a h6 (en este caso a h3)

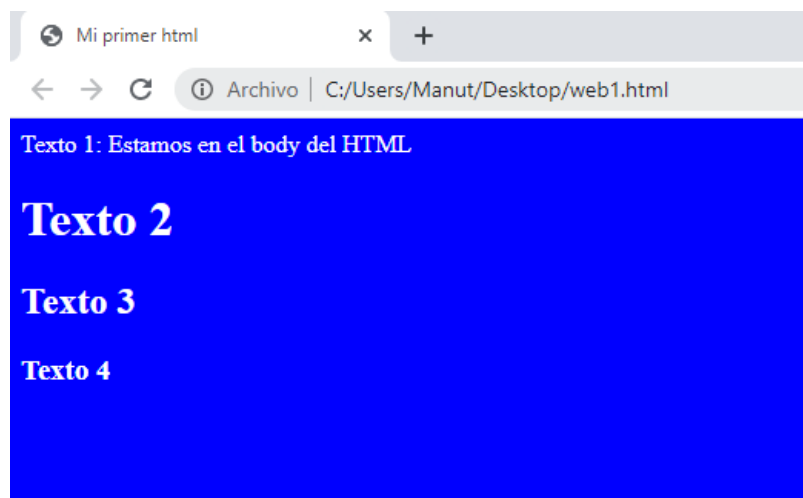


Figura 1.9: Resultado

tr y td

Podríamos crear una tabla si recordamos que "table" es la instrucción, y sabiendo que "tr" sirve para añadir una fila y "td" una columna de la siguiente manera.

Veremos como al escribir un <tr> le decimos que es una fila, y dentro tenemos <td>, tantos como columnas queramos. borde de la tabla ("border = 7")

Para hacer el ejemplo y que se vea claramente añadiremos 3 filas y 3 columnas

```
<body bgcolor="blue" text="white">
  <p>Texto 1: Estamos en el body del HTML</p>
  <table border=7>
    <tr>
      <td>Fila 1 - Columna 1</td>
      <td>Fila 1 - Columna 2</td>
      <td>Fila 1 - Columna 3</td>
    </tr>
    <tr>
      <td>Fila 2 - Columna 1</td>
      <td>Fila 2 - Columna 2</td>
      <td>Fila 2 - Columna 3</td>
    </tr>
    <tr>
      <td>Fila 3 - Columna 1</td>
      <td>Fila 3 - Columna 2</td>
      <td>Fila 3 - Columna 3</td>
    </tr>
  </table>
</body>
```

Figura 1.10: tr y td

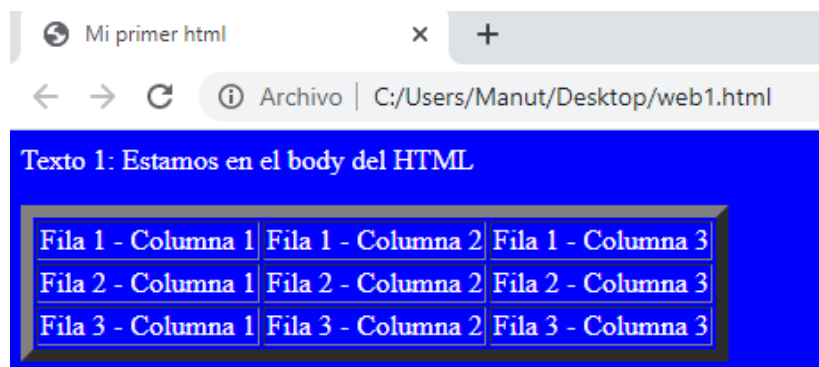


Figura 1.11: Resultado de tr y td

center

```
<html>
  <head>
    <title>
      Mi primer html
    </title>
  </head>

  <body bgcolor="grey" text="white">
    <center>
      <p>Texto 1: Estamos en el body del HTML</p>
    </center>
  </body>
</html>
```

Figura 1.12: Center para centrar un texto en este caso

Hemos modificado el color, y simplemente centrado el texto inicial con la etiqueta `<center>.....</center>`

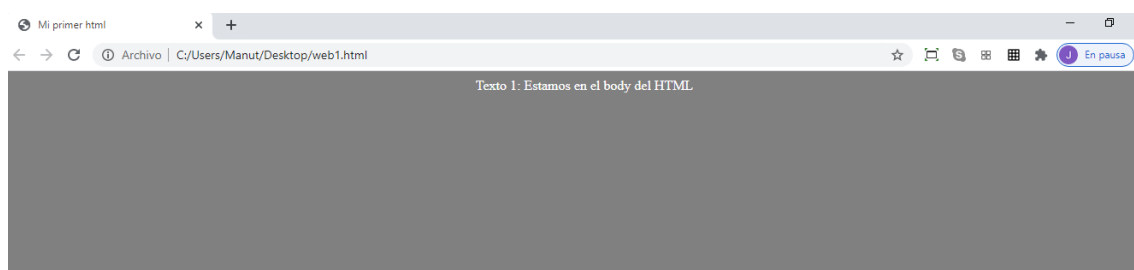


Figura 1.13: Texto centrado

Siempre que queremos centrar algo trataremos de hacerlo de esta forma.

Link a una web o a una url

Algo que utilizaremos en alguna ocasión (con Django) es el "href",

Para poder explicar un poco como funciona haremos el siguiente código ejemplo.

```
1  <html>
2      <head>
3          <title>
4              Mi primer html
5          </title>
6      </head>
7      <body bgcolor="grey" text="white">
8          <a href="https://eiposgrados.edu.es/">Escuela de Posgrados</a>
9      </body>
10 </html>
```

Figura 1.14: Link a la Escuela Internacional de Posgrados

Que quedaría algo tal que así:

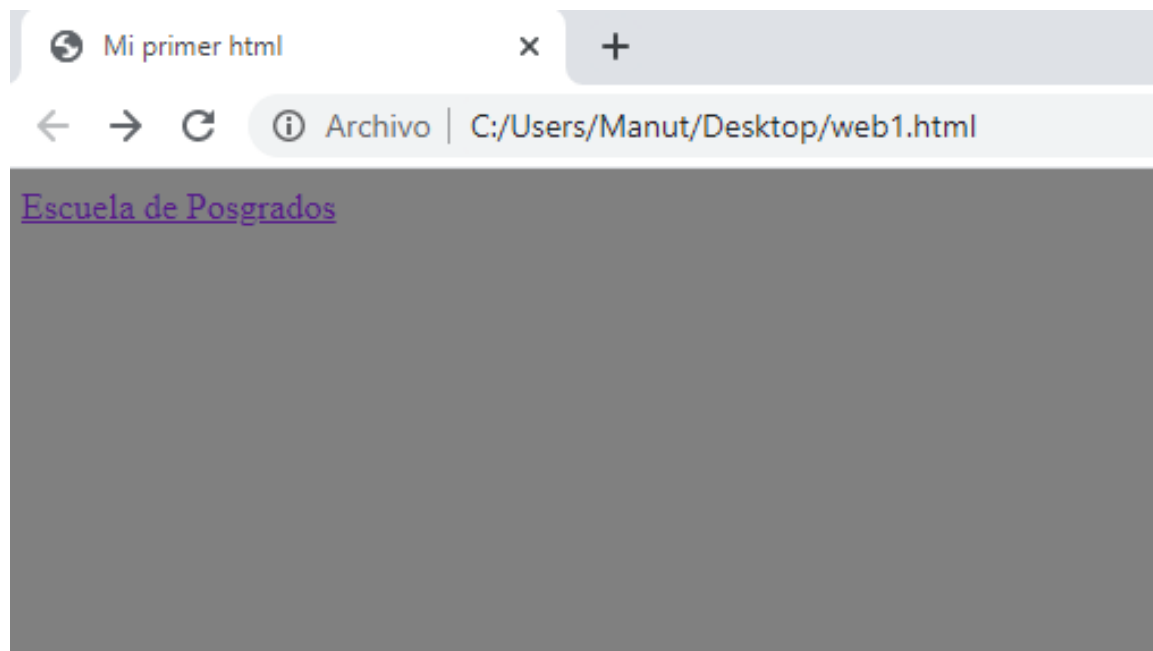


Figura 1.15: href, link a una web

Si hacemos doble click en “Escuela de Posgrados” nos llevará directamente a la web.



Figura 1.16: Web de la Escuela tras hacer click

2. CSS3: BREVÍSIMA INTRODUCCIÓN

CSS nos va a permitir mejorar la apariencia.

Su nombre viene de *Cascading Style Sheets* (*Hojas de Estilo en Cascada*).

Fueron desarrolladas por W3C. El objetivo es complementar al HTML.

Para explicarlo crearemos un nuevo archivo html (web2.html) en mi caso.

Y crearemos otro archivo, un archivo .css (stylesheet.css en mi caso)

Una vez en el CSS, nuestro Entorno de Desarrollo ATOM nos permite buscar varias opciones a cada etiqueta de la siguiente forma

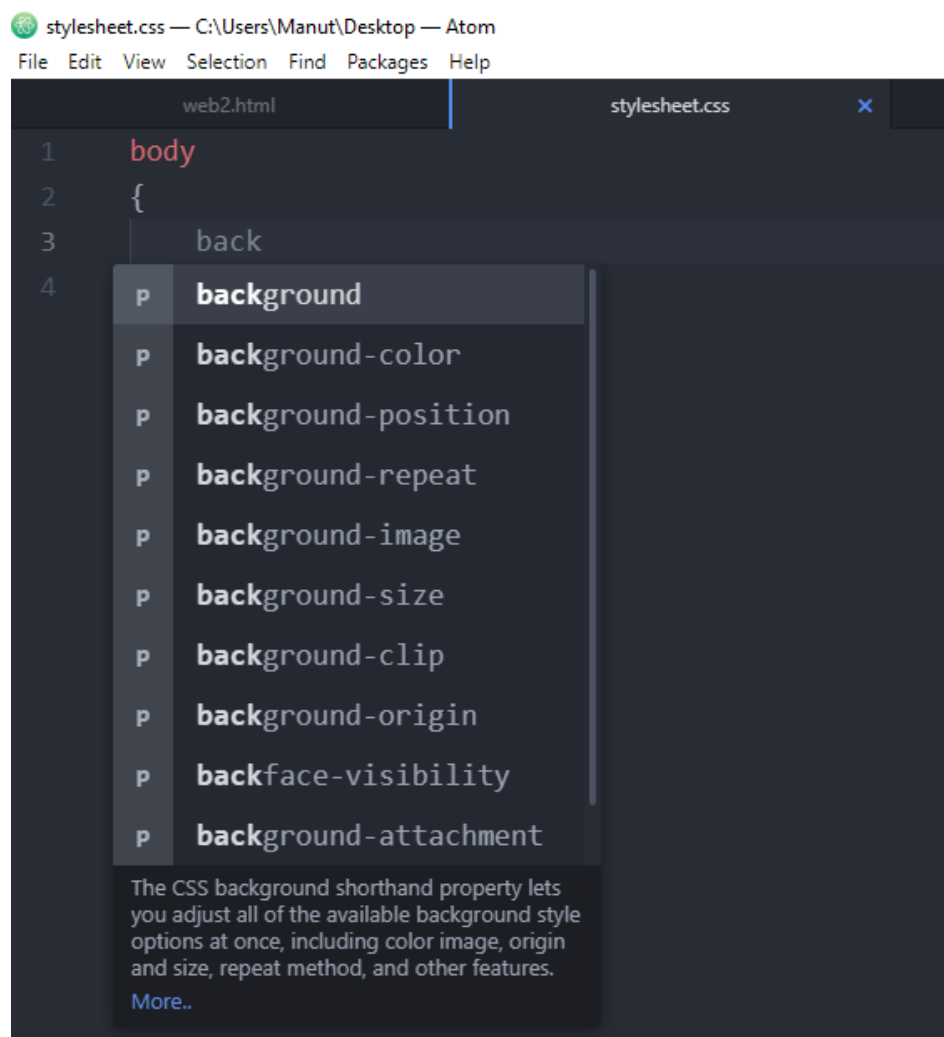


Figura 2.1: CSS para el body

<https://www.color-hex.com/color-wheel/>

en esta web nos permite buscar el mejor color, para que sea más visual.

También pudimos usarlo en HTML, pero lo usaremos de esta forma, con el CSS.

Color Wheel

Select any color your want by clicking on the **color wheel** tool, each time you select a color, hex color code will displayed in the box.



Figura 2.2: Color Wheel

También podemos modificar otros parámetros si quisiéramos, aunque en nuestro caso modificaremos únicamente el color de fondo.

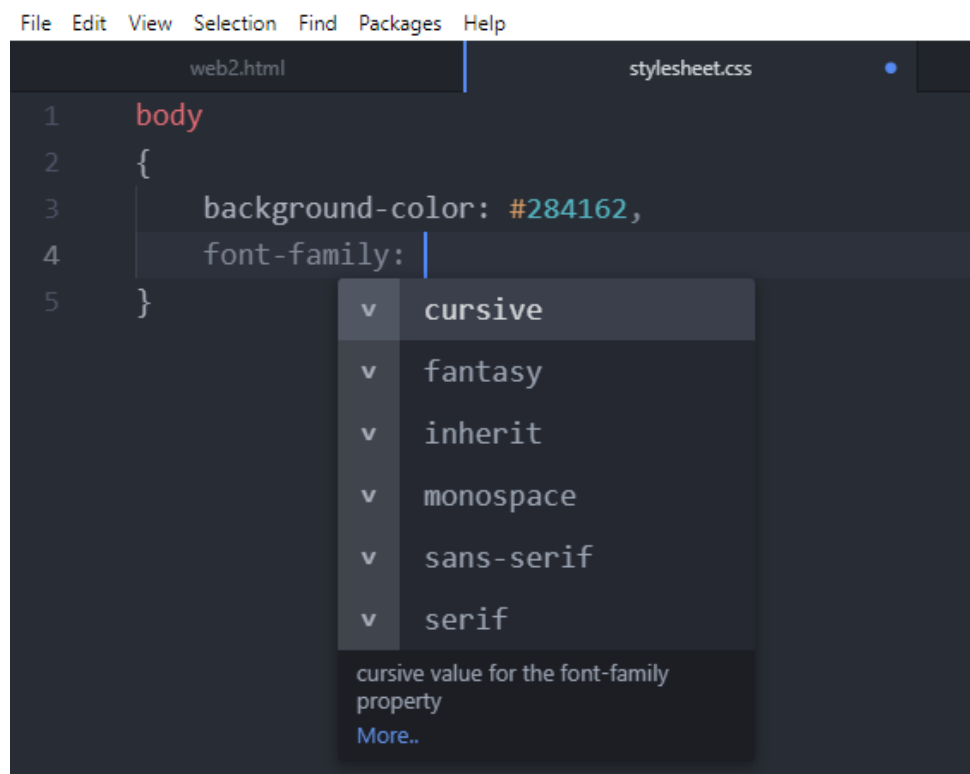


Figura 2.3: Font-family en el body

Y lo que haremos será modificar el color de h1 por ejemplo.

El código podría quedarnos de la siguiente manera

HTML primeramente y CSS a continuación:



```

web2.html — C:\Users\Manut\Desktop — Atom
File Edit View Selection Find Packages Help

web2.html
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <meta charset="utf-8">
5     <title>HTML5 + CSS3</title>
6     <link rel = "stylesheet" type="text/css" href="C:/Users/Manut/Desktop/styles.css">
7   </head>
8   <body>
9     <h1>El tamaño mas grande de h</h1>
10    <h2>El segundo mayor tamaño</h2>
11    <h3>Un poco mas pequeño</h3>
12  </body>
13 </html>
  
```

Figura 2.4: HTML por un lado



```

styles.css — C:\Users\Manut\Desktop — Atom
File Edit View Selection Find Packages Help

web2.html styles.css
1 body
2 {
3   background-color: #284162;
4 }
5 h1
6 {
7   font-family: cursive;
8   color: #f3f5f6;
9 }
10 h2
11 {
12   font-family: sans-serif;
13   color: #7ac9f0;
14 }
15 h3
16 {
17   font-family: monospace;
18 }
  
```

Figura 2.5: CSS por otro lado

Donde lo que vemos nuevo es el <link rel..> que lo que hace es relacionar el HTML con el CSS que explicaremos a continuación.

Lo que haremos ahora será ir a cada elemento que queramos modificar y seguiremos una estructura muy simple, tal y como hemos visto.

Resulta conveniente comentar que cuando el archivo .html y archivo .css se encuentran en la misma carpeta NO es necesario añadir la ruta.

Para el código el resultado de la misma será:

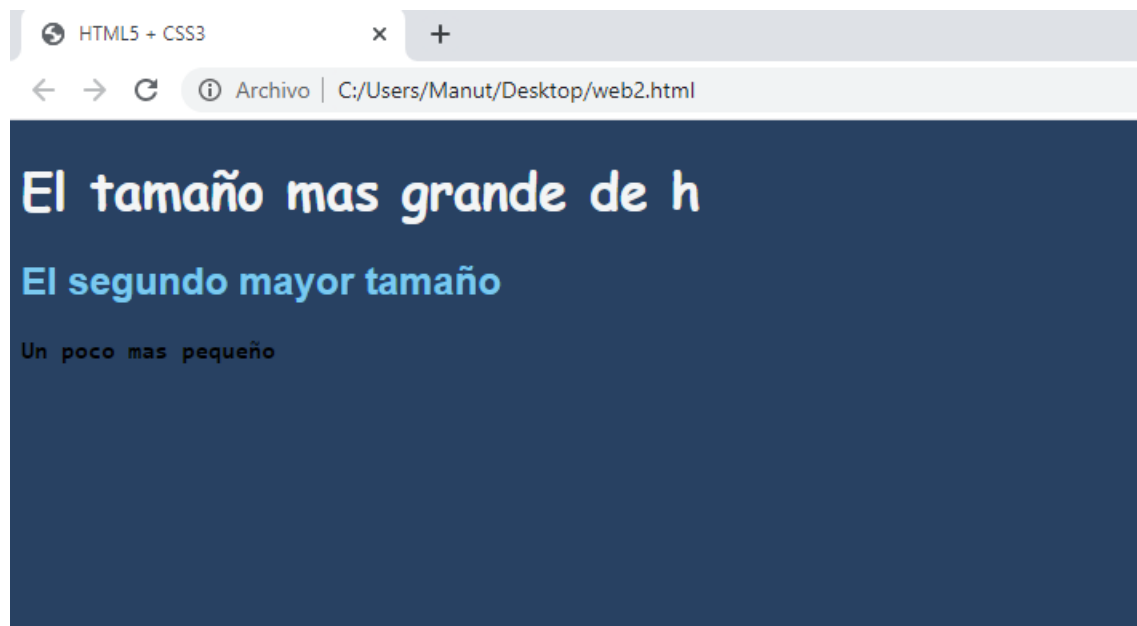


Figura 2.6: HTML+CSS

Existen muchas más opciones, pero por el momento es suficiente.

3. PLANTILLAS DJANGO

base_generic.html

```
base_generic.html
1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4     {% block title %}<title>Django Course</title>{% endblock %}
5     <meta charset="utf-8">
6     <meta name="viewport" content="width=device-width, initial-scale=1">
7     <link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/css/bootstrap.min.css">
8     <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.12.4/jquery.min.js"></script>
9     <script src="https://maxcdn.bootstrapcdn.com/bootstrap/3.3.7/js/bootstrap.min.js"></script>
10
11     <!-- Add additional CSS in static file -->
12     {% load static %}
13     <link rel="stylesheet" href="{% static 'css/styles.css' %}">
14 </head>
```

Figura 3.1: base_generic.html parte 1

Aclaraciones de la Figura 3.1:

- | Idioma "English"(En), inglés.
- | El título va en un "block"
- | UTF-8, sirve para evitar problemas con los caracteres.
- | Hace uso de Bootstrap, que es un Framework de CSS
- | Usa también jquery (JavaScript) y nuevamente Bootstrap, pero nosotros no necesitamos programar JavaScript, Bootstrap si lo usaremos.
- | link para llamar al CSS en la ruta del proyecto Django.

```

16 <body>
17     {% load i18n %}
18     <table align="right" style="margin-right: 25px; margin-top: 20px">
19         <tbody>
20             {% if user.is_authenticated %}
21                 <tr>
22                     <td>{% trans "User:" %} {{ user.username|default:'Guest' }}</td>
23                 </tr>
24                 <tr>
25                     <td><a href="{% url 'logout' %}?next={{ request.path }}">{% trans "Logout" %}</a><td>
26                     <td><a href="{% url 'password_change' %}">{% trans "Change password" %}</a></td>
27                 </tr>
28             {% else %}
29                 <tr>
30                     <td>{% trans "User:" %} {{ user.username|default:'Guest' }}</td>
31                 </tr>
32                 <tr>
33                     <td><a href="{% url 'login' %}?next={{ request.path }}">{% trans "Login" %}</a></td>
34                 </tr>
35             {% endif %}
36         </tbody>
37     </table>

```

Figura 3.2: base_generic.html parte 2

Aclaraciones de la Figura 3.2:

- | Lleva una tabla la cual tiene su body (tbody)
- | Tr y td nos hace la tabla por filas y columnas
- | Lleva un bucle If-Else en Lenguaje de Programación Jinja2
- | Si el usuario está autenticado (user.is_authenticated) te salen 2 botones, 1 de ellos de salida (Logout) y el otro de cambiar contraseña (Change password) Y si no lo está el usuario es invitado (Guest) y el botón de Login (de entrada).

```

38 <div class="container-fluid">
39   <div class="row">
40     <div class="col-sm-2">
41       {% block sidebar %}
42         <ul class="sidebar-nav" style="margin-top: 20px;">
43           <li><a href="{% url 'home' %}">{% trans "Home" %}</a></li>
44           <li><a href="{% url 'iris' %}">{% trans "Iris Data" %}</a></li>
45           <li><a href="{% url 'insertData' %}">{% trans "Insert Data" %}</a></li>
46           <li><a href="{% url 'updateData' %}">{% trans "Update Data" %}</a></li>
47           <li><a href="{% url 'deleteData' %}">{% trans "Delete Data" %}</a></li>
48         </ul>
49       {% endblock %}
50     </div>
51     <div class="col-sm-10">
52       {% block content %}{% endblock %}
53     </div>
54   </div>
55 </div>
56 </body>
57 </html>

```

Figura 3.3: base_generic.html parte 3

Aclaraciones de la Figura 3.3:

- | Es un div, similar a <table>
- | Generamos un menú con una serie de viñetas.
- | sirve para los puntos (viñetas),
 - Tal y como se encuentra esta línea.
- | Después tenemos una segunda división (otro div) donde alojaremos el resto de plantillas

A continuación explicaremos main.html

main.html

```

1  {% extends "base_generic.html" %}
2  {% block content %}

```

Figura 3.4: main.html parte 1

Aclaraciones de la Figura 3.4:

- | Lo que estamos diciendo es que se va a alojar en la base_generic.html
- | Y abrimos el block.

```

3  <html>
4      <body>
5          {% load i18n %}
6          <h1>{% trans "Iris Dataset" %}</h1>
7          <br>
8          <h3>{% trans ". Resume:" %}</h3>
9          <table class="table table-hover">
10             <tr>
11                 <td></td>
12                 <td><label>{% trans "Sepal Length" %}</label></td>
13                 <td><label>{% trans "Sepal Width" %}</label></td>
14                 <td><label>{% trans "Petal Length" %}</label></td>
15                 <td><label>{% trans "Petal Width" %}</label></td>
16             </tr>
17             {% for key, value in describe.items %}
18                 <tr>
19                     <td><strong>{{ key }}</strong></td>
20                     <td>{{ value.sepal_length }}</td>
21                     <td>{{ value.sepal_width }}</td>
22                     <td>{{ value.petal_length }}</td>
23                     <td>{{ value.petal_width }}</td>
24                 </tr>
25             {% endfor %}
26         </table>

```

Figura 3.5: main.html parte 2

Aclaraciones de la Figura 3.5:

- | Tiene un título h1 y un título h3.
- | El br es un salto de línea.
- | Tiene una tabla con un modelo Bootstrap (class=table table-hoover) que es un diseño concreto.
- | Tiene un tr con varios td.
- | Y después un bucle for en Jinja2 en el cual nos mostrará el contenido de la tabla.

```

27     <br>
28     <h3>{% trans ". All Data:" %}</h3>
29     <table class="table table-hover">
30         <tr>
31             <td></td>
32             <td><label>{% trans "Sepal Length" %}</label></td>
33             <td><label>{% trans "Sepal Width" %}</label></td>
34             <td><label>{% trans "Petal Length" %}</label></td>
35             <td><label>{% trans "Petal Width" %}</label></td>
36             <td><label>{% trans "Species"%}</label></td>
37         </tr>
38         {% for key, value in data.items %}
39             <tr>
40                 <td><strong>{{ key }}</strong></td>
41                 <td>{{ value.sepal_length }}</td>
42                 <td>{{ value.sepal_width }}</td>
43                 <td>{{ value.petal_length }}</td>
44                 <td>{{ value.petal_width }}</td>
45                 <td>{{ value.species }}</td>
46             </tr>
47         {% endfor %}
48     </table>
49 </body>
50 </html>

```

Figura 3.6: main.html parte 3 (similar a la anterior)

```

51     {% endblock %}

```

Figura 3.7: main.html parte 4 (cierro el "block")

insert.html

```

1  {% extends "base_generic.html" %}
2  {% block content %}
3  <html>
4  <body>
5      {% load i18n %}
6  <form method='post'>
7      {% csrf_token %}
8      <h1>{% trans "Iris Dataset" %}</h1>
9      <br>
10     <h3>{% trans "· Insert Data:" %}</h3>
11     {% if result != '' %}
12         <label style="color:green;">{{ result }}</label>
13     {% endif %}

```

Figura 3.8: insert.html parte 1

Aclaraciones de la Figura 3.8:

- | Va alojada en base_generic.html
- | Abrimos el block
- | Y tenemos un body y vamos a crear un formulario con método "post".

{% csrf_token %} es el token de seguridad del método post, que veremos en la siguiente lección con un poco más de detalle

- | Tiene 2 títulos h1 y h3, con un espacio.
- | Lo siguiente es un código nuevamente Jinja2 que es un bucle if que cuando el resultado es distinto de vacío " " muestre un label.

```

14 <table class="table table-hover">
15   <tr>
16     <th><label>{% trans "Sepal Length:" %}</label></th>
17     <td><input name="sepal_length" value="" type="number" step='0.1' /></td>
18   </tr>
19   <tr>
20     <th><label>{% trans "Sepal Width:" %}</label></th>
21     <td><input name="sepal_width" value="" type="number" step='0.1' /></td>
22   </tr>
23   <tr>
24     <th><label>{% trans "Petal Length:" %}</label></th>
25     <td><input name="petal_length" value="" type="number" step='0.1' /></td>
26   </tr>
27   <tr>
28     <th><label>{% trans "Petal Width:" %}</label></th>
29     <td><input name="petal_width" value="" type="number" step='0.1' /></td>
30   </tr>
31   <tr>
32     <th><label>{% trans "Species:" %}</label></th>
33     <td><input name="species" value="" type="text" /></td>
34   </tr>
35 </table>

```

Figura 3.9: insert.html parte 2

Aclaraciones de la Figura 3.9:

- Es una tabla en la cual vemos tr, td y th, donde th es otra opción que tenemos.

```

36 <input type="submit" name="Submit">
37 </form>
38 </body>
39 </html>
40 {% endblock %}

```

Figura 3.10: insert.html parte 3

Aclaraciones de la Figura 3.10:

- input es el botón de enviar
- Y cerramos el formulario, cerramos el body el html y el block.

styles.css

```
1  ✓  .sidebar-nav {  
2      margin-top: 20px;  
3      padding: 0;  
4      list-style: none;  
5  }
```

Figura 3.11: styles.css

Aclaraciones de la Figura 3.11:

- | Es el formato del menú escrito en código CSS.
- | Margen en la parte superior 20 píxeles
- | Padding es 0
- | Y estilo de lista no tiene.

4. PUNTOS CLAVES

- | Conocer Frameworks de Python para hacer Aplicaciones Webs nos ahorrará muchísimo tiempo, pero, no olvides que siempre será útil tener unos conocimientos de HTML y CSS para mejorar el diseño de algunos proyectos e incluso para entender las plantillas que suelen usarse en Frameworks como Django.

